Neural Style Transfer of Different Art Styles

Ayçe İdil Aytekin (21803718)
Cemal Gündüz (21703004)
Eren Şenoğlu (21702079)
Dora Tütüncü (21801687)

CS464 Term Project
Group 8
May 14, 2021

Contents:

INTRODUCTION

In this project, we intended to propose a solution to the artistic style transfer context which modifies an image with respect to the style of another image. In order to achieve this input images are divided into two subclasses as ‘style’ images and ‘content’ images. The generated output will possess the content information of the ‘content’ image and have the style of the ‘style’ image. As the first milestone, the team decided to implement non-photorealistic style transfer in order to simplify the complexity of the image stylization problem.

There exist different ways of achieving non-photorealistic image stylization. One of them is the utilization of the Convolutional Neural Networks (CNN) and the other one is the Generative Adversarial Networks (GAN). Considering the complexity and precision of both methodologies, the team decided to implement VGG19 (CNN) first, CycleGAN second.

In the results, both methodologies successfully transferred the style and achieved the aim of the project. A detailed discussion of the results can be found in the results and conclusion part.

PROBLEM DESCRIPTION

- How do we maintain the content of the image while its style is changing with respect to another image?

The problem addressed in this project is to obtain an image, such as a photograph, stylized in the form of another image, such as a painting. Of course, the problem here is to obtain an image such that the originality of the “content” image is preserved such that it is recognizable, however, it appears as if it were the objects in the “Starry Night” painting of Van Gogh or was in the style of an impressionist era painting done by the likes of Claude Monet. In order to address this problem, the content and style images must be modeled such that their distinguishing features are established while keeping the losses in the style and content to a minimum.

- How can we obtain the most accurate output images in the neural style-transfer process?

There are different approaches to neural style transfer problems, so we need to find and choose the best solution with the best results according to given datasets. That's why we tried to use VGG-19 in the first part of the project and then we used CycleGAN in the second part. To see their differences in style, required times to train, and the output performances.

METHODS

1) VGG19

The first method we used for style transfer was through the implementation of the VGG-19 Convolutional Neural Network (CNN), which is a network pre-trained on the ImageNet dataset. VGG-19 has a total of 19 layers consisting of 16 convolutional layers, 3 fully connected layers, 5 max-pooling layers, and 1 softmax layer. In our implementation, following Gatys' paper, we did not use fully connected layers and average pooling was used instead of max-pooling [1]. When an input image is fed into VGG19, activations of the first layers recognize the shades or edges of the input image, while activations of later layers recognize more advanced features such as textures, shapes, background objects, and patterns. It must be noted that Since the parameters and the weights of the VGG19 are not changed, they are set to their values during CNN initialization. In order to begin the style transfer process, a copy of the content image is initially fed to the system to later be changed according to the calculated loss.

The approach to the style transfer problem using CNN involves two loss functions defined as the content and style loss. The total loss function that will be minimized in order to establish style transfer is as given in figure 1;

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x}) \quad (1)$$

Figure 1: Total Loss Function

where "a" indicates the style image," p" indicates the content image and "x" is the resulting image. α and β are the weighting factors for content and style reconstruction respectively. We chose α as 1 and β as 10^3 following Gatys' paper which says the ratio of α over β should be 10^{-3}.

For this method, a content and style image in the context of the desired style transfer must be used. First, both images were preprocessed. The ImageNet mean is subtracted from the images so that before training, images are normalized. Since VGG19 was pre-trained using Caffe and images were loaded in VGG19 using OpenCV, which loads images in BGR format, VGG19 expects inputs to be in BGR format; so, images are turned to BGR from RGB during preprocessing. Then, the images are turned into tensors.

Next, the content information of a content image is captured by a feature map in a layer "l" in the CNN. The responses in a layer "l" can be stored in a matrix F^l, which is a matrix of real numbers in the dimensions equal to the number of features maps and its size, where F^l_{ij} is the activation of the i^{th} filter at position j in layer "l". To visualize the image information encoded at different layers of the CNN, we let p be the original image vector and x the image that is generated with P^l and F^l their respective feature representation in layer "l". We then defined the squared-error loss between the two feature representations as in figure 2 below.

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$

Figure 2: Content Loss Function

On top of the CNN responses in each layer of the network, we built a style representation that computes the correlations between the different filter responses, where the expectation is taken over the spatial extent of the input image. These feature correlations are given by the Gram matrix. The style cost of a hidden layer "l" can be calculated by describing the style as a correlation between activations across (BGR) channels. Specifically, the style matrices (gram matrix) as unnormalized cross-covariance for style and context images are as in figure 3:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

Figure 3: Gram Matrix

where F denotes the vectorized feature map i and j in layer l.

Lastly, the contribution of a single layer to total loss (fig. 4) and total style loss (fig. 5) can be expressed as:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l}\right)^2$$

Figure 4: Error Function

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Figure 5: Style Loss Function

where w_l is the weighting factor of the contribution of each layer to the total loss. w_l is taken as ⅕ for style loss layers where w_l is taken as 1 for content loss layer.

Having established the style loss in figure 5 and the content loss in figure 2, we weigh them as in equation 1 according to their corresponding values and minimize error. Having established the mathematical error representations, the learning and gradient descent process can start.

To begin the gradient descent, the gradients of each loss must be obtained with respect to each activation layer, which can be done in the standard error back-propagation using the Pytorch 'backwards' function. It must be noted that for the content image, the "conv4_2" in the VGG19 layer is used, while for the style reconstructions and losses, "conv1_1", "conv2_1", "conv3_1", "conv4_1", "conv5_1" layers are used.

CNN's must use an optimizer in the training process. The first candidate optimizer to be used was the LBFGS. Another candidate optimizer was Adam, but since inputs are 2 static images, Adam does not work well in style transfer. In addition to this, LBFGS gave faster and more reliable results, so LBFGS was used.

VGG19 Dataset: As style images, famous paintings from famous artists are used, and as content images, random images were selected from Google Images.

2) CycleGAN

CycleGAN is a type of Generative Adversarial Network that is used for unpaired image-to-image translation. A typical GAN structure has one generator and one discriminator whereas CycleGAN has 2 generators and 2 discriminators. Let X be the source image domain and Y be the target image domain. Generator G tries to learn the mapping from X to Y whereas generator F tries to learn the mapping from Y to X. The discriminator Dx tries to distinguish between real source images from domain X and fake source images generated by F. The discriminator Dy tries to distinguish between real target images from domain Y and fake target images generated by G. The figure below [2] represents these relations.

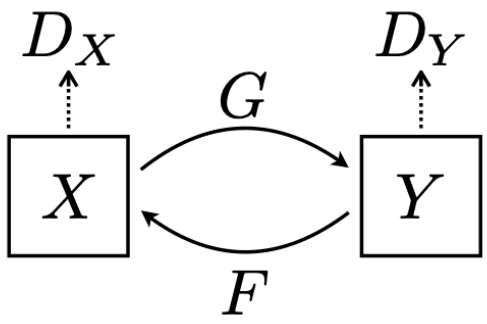


Figure 6: Generator and Discriminator Scheme

In CycleGAN, we try to solve

$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_x, D_y)$$

Figure 7: Generator and Discriminator Formula

There are 3 types of losses in CycleGAN algorithm:

1. **Adversarial Loss**: Adversarial loss belongs to the GAN structures. G tries to produce similar images to Y while D_y tries to distinguish real Y from the fake Y which is G(X). F tries to produce similar images to X while D_x tries to distinguish real X from the fake X which is F(Y). This main objective can be expressed as

$$\min_{G} \max_{D_y} \mathcal{L}_{GAN}(G, D_y, X, Y)$$

Figure 8: Adversarial Loss Target Formula

where Dy tries to maximize the loss so that it can distinguish and G tries to minimize the loss so that it can produce similar images. The adversarial loss can be written as

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{y \sim p_{data}(x)}[\log (1 - D_Y(G(x))] + \mathbb{E}_{y \sim p_{data}(x)}[\log D_X(y)] + \mathbb{E}_{y \sim p_{data}(y)}[\log (1 - D_X(F(y))]$$

Figure 9: Adversarial Loss Formula

2. **Cycle Consistency Loss:** Cycle consistency loss comes from when an output from G, G(X) is given to F as an input, the output of F, F(G(X)) should be matching to X. Similar happens for the reverse. It is an L1 norm loss.

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{y \sim p_{data}(x)}\left[\left\|F(G(x)) - x\right\|_1\right] + \mathbb{E}_{y \sim p_{data}(y)}\left[\left\|G(F(y)) - y\right\|_1\right]$$

Figure 10: Cyclic Loss

3. **Identity Loss**: Identity loss encourages the mapping to preserve color and composition between the input and output images of the generators. When this loss is not used, the tinting might arbitrarily change. Identity loss can be expressed as

$$\mathcal{L}_{identity}(G, F) = \mathbb{E}_{y \sim p_{data}(x)}[\|F(x) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)}[\|G(y) - y\|_1]$$

Figure 11: Identity Loss

Total loss is the sum of the adversarial loss, cycle consistency loss and identity loss. It can be written as

$$\mathcal{L}(G, F, D_x, D_y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda_1 \mathcal{L}_{cyc}(G, F) + \lambda_2 \mathcal{L}_{identity}(G, F)$$

Figure 12: Total Loss

According to [2], λ_1 is chosen as 10 and λ_2 is chosen as 5.

CycleGAN consists of 2 main parts: Generator block and Discriminator block. We use Generators to produce photos and paintings. So, our X domain is photographs and our Y domain is paintings, specifically Van Gogh's. In the generator structure, images are first downsampled with 2 convolutional layers and then processed through Residual blocks which are described later. Then, to arrive at the images' original dimensions, upsampling is used.

Residual Blocks are used to decrease the training errors because, without them, training errors start to increase at some point as more layers are added to the neural network.

The Discriminator consists of 70 by 70 PatchGANs. It tries to distinguish whether the overlapping patch is fake or real. As not the whole image size is used, PatchGANs have fewer parameters and flexible input size. Thus, they are less computationally expensive. BatchNormalization is used instead of InstanceNormalization so that we can standardize each output feature map instead of whole features in a batch. In the end, average pooling is applied to acquire one-dimensional labels and the output is a probability map that shows each region of the image is fake or real.

In addition, a structure called Replay Buffer is used to train the discriminator. It works in the following fashion: if the buffer is not full, insert images; if the buffer is full, return a previously stored image and put the current image into the buffer with %50 probability or return the current image with %50 probability. This way, the oscillation of the model can be reduced.

The learning rate is 0.0002 for 100 epochs and for the rest 100 epochs, it linearly decreases to 0. Adam is used as the optimizer and its beta parameters are (0.5, 0.999) [2]. The network is trained for 200 epochs with batch size 1 and a batch sampler for faster training.

CycleGAN Dataset: In this part, we used Van Gogh's paintings as style and landscape photos used as content images. For this purpose, we use a dataset named "vangogh2photo" which includes 7.8k samples in it from CycleGAN's original implementation. Samples are divided into four groups and two classes. First class Van Gogh's paintings and the other class is real landscape photos. These classes are divided into a total of four groups as train and test. As preprocessing, images are normalized and augmented to possess the correct shape for the algorithm.

RESULTS

VGG19: It took almost 1 minute to acquire the style transferred image. The results are in Appendix B. The content of the image is not thoroughly preserved but the style is transformed. According to the α and β, the effect of the content image and style image in the resulting image can be arranged. When α was increased, the content image's effect increased and when β was increased, the effect of the style image increased.

CycleGAN: At first, when the batch size was 1 and the epoch number was 200, it took more than 6 hours and Google Colab collapsed. Then, we used a batch sampler and chose batch size 1 and epoch 200 again, this time it took approximately 40 minutes. The results are as in Appendix C. The content of the images were preserved but styles could not be transformed successfully due to an insufficient amount of training epochs.

DISCUSSION

It can be seen that VGG19 results are better than CycleGAN results in terms of change in the style while maintaining the content. VGG19 results seem more realistic and aesthetically pleasing. In the beginning, we thought that we could get better results if we used CycleGAN, however, output images appeared worse than VGG19. The biggest problem was CycleGAN requires a high amount of computational power and time. We implemented the project on "Google Colab" and it couldn't satisfy the computational power needed by CycleGAN. It took a lot more time than VGG19 in each epoch and finally Google Collab gave an error after like 6 hours of runtime when we tried to train with more epochs.

CONCLUSION

- How do we maintain the content of the image while its style is changing with respect to another image?

  In VGG19, the earlier layers of the network detect edges and shades (low-level features) while deeper (later) layers detect more advanced features such as shapes, complex textures, background objects. Since we want the resulting image to have similar content as the content image, we have chosen layers that are not too early or too deep. So, the content is maintained.

  In CycleGAN, both content and style are preserved via the adversarial loss, cycle consistency loss and identity loss as described in the Methods part.

- How can we obtain the most accurate output images in the neural style-transfer process?

  Each solution had different approaches and methodologies to the problem of stylization. However, the biggest difference was arising from the required computational power to train the models. In the VGG19 solution, the team utilized the pre-trained model and in the CycleGAN solution, the team trained the model with the limited amount of computational resources they had. Due to this factor, the generated model performed worse than the actual CycleGAN models and the pre-trained VGG19 model.

  We can conclude that the results of the VGG19 model were more accurate than the CycleGAN model. However, by satisfying the needs of the CycleGAN method one can produce better stylizations than the VGG19.

In the future, neural style transfer can be implemented by the Contrastive Unpaired Translation algorithm which is said to be working faster than CycleGAN and consuming less memory when compared to CycleGAN. Its main principle is maximizing the mutual information between the input and output patches of images via contrastive learning [3].

REFERENCES

[1] L. Gatys, A. Ecker, and M. Bethge, “A Neural Algorithm of Artistic Style,” Journal of Vision, vol. 16, no. 12, p. 326, 2016.

[2] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks,” *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[3] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, “Contrastive Learning for Unpaired Image-to-Image Translation,” *Proceedings of (ECCV) European Conference on Computer Vision*, pp. 319–345, Aug. 2020.

APPENDIX

A. Work Division

The work done for this project in the first method can be divided into the following steps for the first method:

- Constructing the VGG19 CNN
- Constructing Content Loss, Gram Matrix, Style Loss
- Image Preprocessing
- Defining Feature Maps
- Weighing Losses
- Training Loop
- Resulting Image Postprocessing

The work done for this project in the second method can be divided into the following steps:

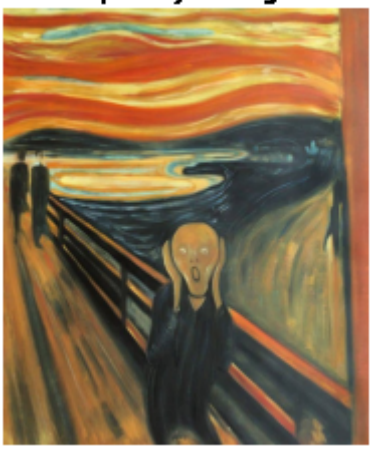
- Data Augmentation
- Image Preprocessing
- Constructing Batch Sampler
- Constructing Residual Block Structure
- Constructing Generator
- Constructing Discriminator
- Constructing Replay Buffer
- Adjusting Learning Rate with LambdaLR Scheduler
- Constructing Optimizers
- Training the CycleGAN network
- Inferring Test Images with the Trained Network (Evaluating the Model)

All parts were implemented in the pair programming manner.

B. Results of VGG-19

B.1.1. Produced Images from Style Transfer of the "Scream" painting to the "Mona Lisa" painting

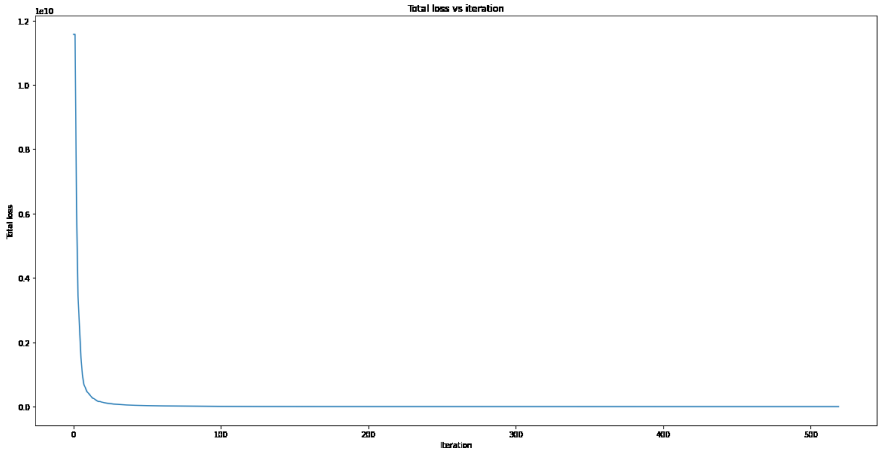
Input Style Image



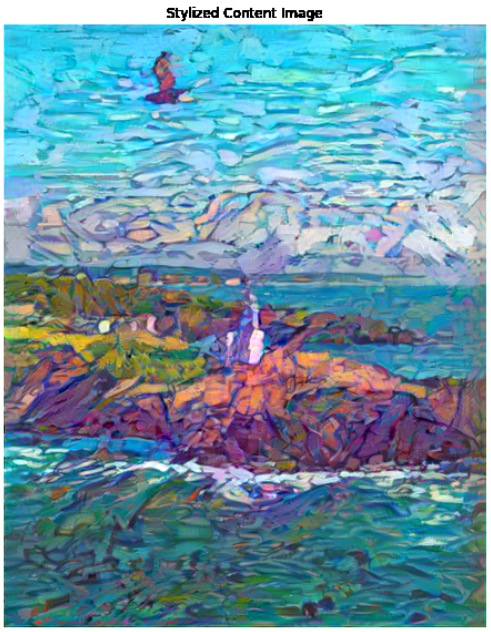
Input Content Image

Stylized Content Image

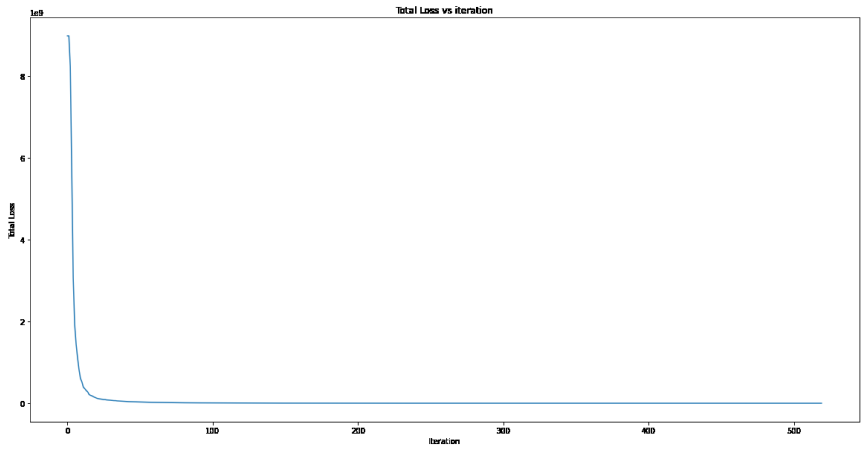
B.1.2. Loss Graph from the Style Transfer of the “Scream” painting to the “Mona Lisa” painting



B.2.1. Produced Images from the Style Transfer of an Erin Hanson painting to a photograph of Sivas İnceburun



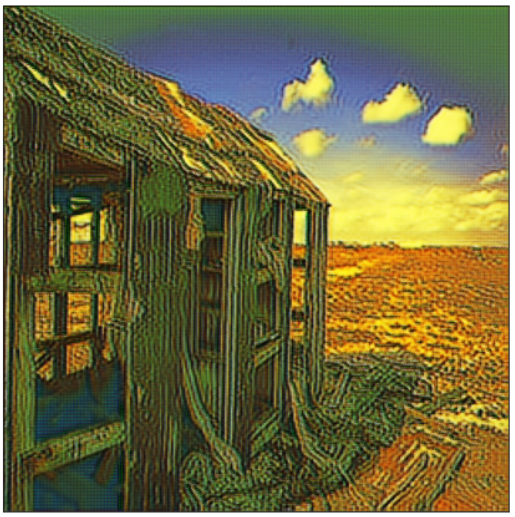
B.2.2. Loss Graph from the Style Transfer of an Erin Hanson painting to a photograph of Sivas İnceburun



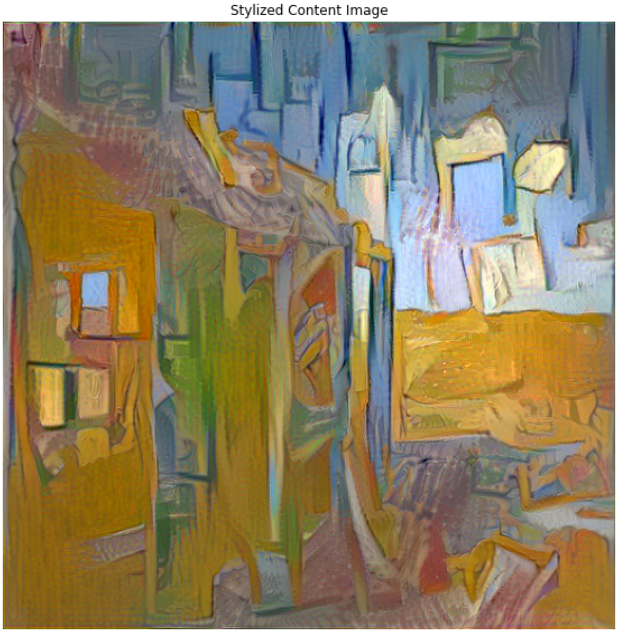
C. Results of CycleGAN

C.1. CycleGAN results for the transfer of a picture of the Earth to the Van Gogh style

C.2.1. CycleGAN results for the transfer of a picture of an abandoned shack to the Van Gogh style



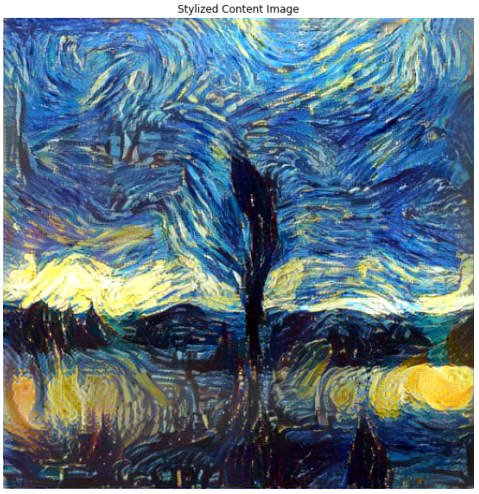
C.2.2. Comparison of the same image with a Van Gogh painting with a stylized image using VGG19



C.3.1. CycleGAN results for the transfer of a picture of a tree to the Van Gogh style

C.3.2. Comparison with the VGG19 version of the same image stylized using the "Starry Night"

Stylized Content Image



D. CycleGAN results for a fully trained model [2]